1 Static Semantics

$\boxed{\Theta; \Delta; \Gamma \vdash e : t}$ Typing rules for expressions

$$\frac{}{\Theta; \Delta; \cdot, x : t \vdash x : t} \quad \textsc{Ty\_Var\_Lin}$$

$$\frac{x : t \in \Delta}{\Theta; \Delta; \cdot \vdash x : t} \quad \textsc{Ty\_Var}$$

$$\frac{\begin{array}{l}\Theta; \Delta; \Gamma \vdash e : t \\ \Theta; \Delta; \Gamma', x : t \vdash e' : t'\end{array}}{\Theta; \Delta; \Gamma, \Gamma' \vdash \mathbf{let}\, x = e\, \mathbf{in}\, e' : t'} \quad \textsc{Ty\_Let}$$

$$\frac{}{\Theta; \Delta; \cdot \vdash () : \mathbf{unit}} \quad \textsc{Ty\_Unit\_Intro}$$

$$\frac{\begin{array}{l}\Theta; \Delta; \Gamma \vdash e : \mathbf{unit} \\ \Theta; \Delta; \Gamma' \vdash e' : t\end{array}}{\Theta; \Delta; \Gamma, \Gamma' \vdash \mathbf{let}\, () = e\, \mathbf{in}\, e' : t} \quad \textsc{Ty\_Unit\_Elim}$$

$$\frac{}{\Theta; \Delta; \cdot \vdash \mathbf{true} : \mathbf{bool}} \quad \textsc{Ty\_Bool\_True}$$

$$\frac{}{\Theta; \Delta; \cdot \vdash \mathbf{false} : \mathbf{bool}} \quad \textsc{Ty\_Bool\_False}$$

$$\frac{\begin{array}{l}\Theta; \Delta; \Gamma \vdash e : !\mathbf{bool} \\ \Theta; \Delta; \Gamma' \vdash e_1 : t' \\ \Theta; \Delta; \Gamma' \vdash e_2 : t'\end{array}}{\Theta; \Delta; \Gamma, \Gamma' \vdash \mathbf{if}\, e\, \mathbf{then}\, e_1\, \mathbf{else}\, e_2 : t} \quad \textsc{Ty\_Bool\_Elim}$$

$$\frac{}{\Theta; \Delta; \cdot \vdash k : \mathbf{int}} \quad \textsc{Ty\_Int\_Intro}$$

$$\frac{}{\Theta; \Delta; \cdot \vdash el : \mathbf{elt}} \quad \textsc{Ty\_Elt\_Intro}$$

$$\frac{\begin{array}{l}\Theta; \Delta; \cdot \vdash v : t \\ v \neq l \cdot f\end{array}}{\Theta; \Delta; \cdot \vdash \mathbf{Many}\, v : !t} \quad \textsc{Ty\_Bang\_Intro}$$

$$\frac{\begin{array}{l}\Theta; \Delta; \Gamma \vdash e : !t \\ \Theta; \Delta, x : t; \Gamma' \vdash e' : t'\end{array}}{\Theta; \Delta; \Gamma, \Gamma' \vdash \mathbf{let}\, \mathbf{Many}\, x = e\, \mathbf{in}\, e' : t'} \quad \textsc{Ty\_Bang\_Elim}$$

$$\frac{\begin{array}{l}\Theta; \Delta; \Gamma \vdash e : t \\ \Theta; \Delta; \Gamma' \vdash e' : t'\end{array}}{\Theta; \Delta; \Gamma, \Gamma' \vdash (e, e') : t \otimes t'} \quad \textsc{Ty\_Pair\_Intro}$$

$$\frac{\begin{array}{l}\Theta; \Delta; \Gamma \vdash e_{12} : t_1 \otimes t_2 \\ \Theta; \Delta; \Gamma', a : t_1, b : t_2 \vdash e : t\end{array}}{\Theta; \Delta; \Gamma, \Gamma' \vdash \mathbf{let}\, (a, b) = e_{12}\, \mathbf{in}\, e : t} \quad \textsc{Ty\_Pair\_Elim}$$

$$\frac{\Theta \vdash t' \ \mathsf{Type} \quad \Theta; \Delta; \Gamma, x : t' \vdash e : t}{\Theta; \Delta; \Gamma \vdash \mathbf{fun}\, x : t' \to e : t' \multimap t} \quad \textsc{Ty\_Lambda}$$

$$\frac{\Theta; \Delta; \Gamma \vdash e : t' \multimap t \quad \Theta; \Delta; \Gamma' \vdash e' : t'}{\Theta; \Delta; \Gamma, \Gamma' \vdash e\, e' : t} \quad \textsc{Ty\_App}$$

$$\frac{\Theta, fc; \Delta; \Gamma \vdash e : t}{\Theta; \Delta; \Gamma \vdash \mathbf{fun}\, fc \to e : \forall fc.t} \quad \textsc{Ty\_Gen}$$

$$\frac{\Theta \vdash f \ \mathsf{Cap} \quad \Theta; \Delta; \Gamma \vdash e : \forall fc.t}{\Theta; \Delta; \Gamma \vdash e[f] : t[f/fc]} \quad \textsc{Ty\_Spc}$$

$$\frac{\Theta; \Delta, g : t \multimap t'; \cdot, x : t \vdash e : t'}{\Theta; \Delta; \cdot \vdash \mathbf{fix}\,(g, x : t, e : t') : t \multimap t'} \quad \textsc{Ty\_Fix}$$

2 Dynamic Semantics

$\boxed{\langle \sigma, e \rangle \to \mathit{StepsTo}}$ operational semantics

$$\frac{}{\langle \sigma, \mathbf{let}\, () = ()\, \mathbf{in}\, e \rangle \to \langle \sigma, e \rangle} \quad \textsc{Op\_Let\_Unit}$$

$$\frac{}{\langle \sigma, \mathbf{let}\, x = v\, \mathbf{in}\, e \rangle \to \langle \sigma, e[x/v] \rangle} \quad \textsc{Op\_Let\_Var}$$

$$\frac{}{\langle \sigma, \mathbf{if}\, (\mathbf{Many\, true})\, \mathbf{then}\, e_1\, \mathbf{else}\, e_2 \rangle \to \langle \sigma, e_1 \rangle} \quad \textsc{Op\_If\_True}$$

$$\frac{}{\langle \sigma, \mathbf{if}\, (\mathbf{Many\, false})\, \mathbf{then}\, e_1\, \mathbf{else}\, e_2 \rangle \to \langle \sigma, e_2 \rangle} \quad \textsc{Op\_If\_False}$$

$$\frac{}{\langle \sigma, \mathbf{let\, Many}\, x = \mathbf{Many}\, v\, \mathbf{in}\, e \rangle \to \langle \sigma, e[x/v] \rangle} \quad \textsc{Op\_Let\_Many}$$

$$\frac{}{\langle \sigma, \mathbf{let}\, (a, b) = (v_1, v_2)\, \mathbf{in}\, e \rangle \to \langle \sigma, e[a/v_1][b/v_2] \rangle} \quad \textsc{Op\_Let\_Pair}$$

$$\frac{}{\langle \sigma, (\mathbf{fun}\, fc \to v)[f] \rangle \to \langle \sigma, v[fc/f] \rangle} \quad \textsc{Op\_Frac\_Cap}$$

$$\frac{}{\langle \sigma, \mathbf{fix}\,(g, x : t, e : t')\, v \rangle \to \langle \sigma, e[x/v][g/\mathbf{fix}\,(g, x : t, e : t')] \rangle} \quad \textsc{Op\_App\_Fix}$$

$$\frac{}{\langle \sigma, (\mathbf{fun}\, x : t \to e)\, v \rangle \to \langle \sigma, e[x/v] \rangle} \quad \textsc{Op\_App\_Lambda}$$

$$\frac{\langle \sigma, e \rangle \to \langle \sigma', e' \rangle}{\langle \sigma, C[e] \rangle \to \langle \sigma, C[e'] \rangle} \quad \textsc{Op\_Context}$$

$$\frac{\langle \sigma, e\rangle \to \mathbf{err}}{\langle \sigma, C[e]\rangle \to \mathbf{err}}\ \textsc{Op\_Context\_Err}$$

$$\frac{0 \le k_1, k_2 \qquad l \text{ fresh}}{\langle \sigma, \mathbf{matrix}\, k_1\, k_2\rangle \to \langle \sigma + \{l \mapsto_1 M_{k_1,k_2}\}, l \cdot 1\rangle}\ \textsc{Op\_Matrix}$$

$$\frac{}{\langle \sigma + \{l \mapsto_1 m_{k_1,k_2}\}, \mathbf{free}\, l \cdot 1\rangle \to \langle \sigma, ()\rangle}\ \textsc{Op\_Free}$$

$$\frac{}{\langle \sigma + \{l \mapsto_f m_{k_1,k_2}\}, \mathbf{share}\, l \cdot f\rangle \to \langle \sigma + \{l \mapsto_{\frac{1}{2}f} m_{k_1,k_2}\} + \{l \mapsto_{\frac{1}{2}f} m_{k_1,k_2}\}, (l \cdot \frac{1}{2}f, l \cdot \frac{1}{2}f)\rangle}\ \textsc{Op\_Share}$$

$$\frac{f \le 1 \qquad v \equiv l \cdot \frac{1}{2}f}{\langle \sigma + \{l \mapsto_{\frac{1}{2}f} m_{k_1,k_2}\} + \{l \mapsto_{\frac{1}{2}f} m_{k_1,k_2}\}, \mathbf{unshare}\, v\, v\rangle \to \langle \sigma + \{l \mapsto_f m_{k_1,k_2}\}, l \cdot f\rangle}\ \textsc{Op\_Unshare\_Eq}$$

$$\frac{l \ne l'}{\langle \sigma + \{l \mapsto_{\frac{1}{2}f} m_{k_1,k_2}\} + \{l' \mapsto_{\frac{1}{2}f} m'_{k_1,k_2}\}, \mathbf{unshare}\, (l \cdot \frac{1}{2}f)\, (l' \cdot \frac{1}{2}f')\rangle \to \mathbf{err}}\ \textsc{Op\_Unshare\_Neq}$$

$$\frac{\begin{array}{l}\sigma' \equiv \sigma + \{l_1 \mapsto_{fc_1} m_{1\,k_1,k_2}\} + \{l_2 \mapsto_{fc_2} m_{2\,k_2,k_3}\} \qquad v_1 \equiv l_1 \cdot f_1 \qquad v_2 \equiv l_2 \cdot f_2 \\ v_3 \equiv l_3 \cdot 1\end{array}}{\langle \sigma' + \{l_3 \mapsto_1 m_{3\,k_1,k_3}\}, \mathbf{gemm}\, v_1\, v_2\, v_3\rangle \to \langle \sigma' + \{l_3 \mapsto_1 (m_1\, m_2 + m_3)_{k_1,k_3}\}, ((v_1, v_2), v_3)\rangle}\ \textsc{Op\_Gemm\_Match}$$

$$\frac{\begin{array}{l}k_2 \ne k_2' \\ \sigma' \equiv \sigma + \{l_1 \mapsto_{fc_1} m_{1\,k_1,k_2}\} + \{l_2 \mapsto_{fc_2} m_{2\,k_2',k_3}\} \\ v_1 \equiv l_1 \cdot f_1 \qquad v_2 \equiv l_2 \cdot f_2 \qquad v_3 \equiv l_3 \cdot 1\end{array}}{\langle \sigma' + \{l_3 \mapsto_1 m_{1\,k_1,k_3}\}, \mathbf{gemm}\, v_1\, v_2\, v_3\rangle \to \mathbf{err}}\ \textsc{Op\_Gemm\_Mismatch}$$

3 Interpretation

3.1 Definitions

Operationally, $\mathit{Heap} \sqsubseteq \mathit{Loc} \times \mathit{Permission} \times \mathit{Matrix}$ (a multiset), denoted with a σ.
Define its *interpretation* to be $\mathit{Loc} \rightharpoonup \mathit{Permission} \times \mathit{Matrix}$ with $\star : \mathit{Heap} \times \mathit{Heap} \rightharpoonup \mathit{Heap}$ as follows:

$$(\varsigma_1 \star \varsigma_2)(l) \equiv \begin{cases} \varsigma_1(l) & \text{if } l \in \mathrm{dom}(\varsigma_1) \wedge l \notin \mathrm{dom}(\varsigma_2) \\ \varsigma_2(l) & \text{if } l \in \mathrm{dom}(\varsigma_2) \wedge l \notin \mathrm{dom}(\varsigma_1) \\ (f_1 + f_2, m) & \text{if } (f_1, m) = \varsigma_1(l) \wedge (f_2, m) = \varsigma_2(l) \wedge f_1 + f_2 \le 1 \\ \text{undefined} & \text{otherwise} \end{cases}$$

Commutativity and associativity of $\star$ follows from that of $+$.
$\varsigma_1 \star \varsigma_2$ is *defined* if it is for all $l \in \mathrm{dom}(\varsigma_1) \cup \mathrm{dom}(\varsigma_2)$.
Implicitly denote $\varsigma \equiv \mathcal{H}[\![\sigma]\!] \equiv \bigstar_{(l,f,m)\in\sigma}[l \mapsto_f m]$.

The $n-$fold iteration for the *StepsTo* (functional) relation, is also a (functional) relation:

$$\forall n.\ \mathbf{err} \to^n \mathbf{err} \qquad \langle \sigma, v\rangle \to^n \langle \sigma, v\rangle \qquad \langle \sigma, e\rangle \to^0 \langle \sigma, e\rangle \qquad \langle \sigma, e\rangle \to^{n+1} ((\langle \sigma, e\rangle \to) \to^n)$$

Hence, all bounded iterations end in either an **err**, a heap-and-expression or a heap-and-value.

3.2 Interpretation

$$\mathcal{V}_k[\![\mathbf{unit}]\!] = \{(\emptyset, *)\}$$

$$\mathcal{V}_k[\![\mathbf{bool}]\!] = \{(\emptyset, true), (\emptyset, false)\}$$

$$\mathcal{V}_k[\![\mathbf{int}]\!] = \{(\emptyset, n) \mid 2^{-63} \leq n \leq 2^{63} - 1\}$$

$$\mathcal{V}_k[\![\mathbf{elt}]\!] = \{(\emptyset, f) \mid f \text{ a IEEE Float64 }\}$$

$$\mathcal{V}_k[\![f\,\mathbf{mat}]\!] = \{(\{l \mapsto_{2^{-f}} \_\}, l)\}$$

$$\mathcal{V}_k[\![!t]\!] = \{(\emptyset, \mathbf{Many}\, v) \mid (\emptyset, v) \in \mathcal{V}_k[\![t]\!]\}$$

$$\mathcal{V}_k[\![\forall fc.\ t]\!] = \{(\varsigma, \mathbf{fun}\, fc \to\ v) \mid \forall f.\ (\varsigma, (\mathbf{fun}\, fc \to\ v)\,[f]) \in \mathcal{V}_k[\![t[fc/f]]\!]\}$$

$$\mathcal{V}_k[\![t_1 \otimes t_2]\!] = \{(\varsigma_1 \star \varsigma_2, (v_1, v_2)) \mid (\varsigma_1, v_1) \in \mathcal{V}_k[\![t_1]\!] \wedge (\varsigma_2, v_2) \in \mathcal{V}_k[\![t_2]\!]\}$$

$$\begin{aligned}\mathcal{V}_k[\![t \multimap t']\!] = \{(\varsigma_{v'}, v') \mid (&v' \equiv \mathbf{fun}\, x : t \to e \vee v' \equiv \mathbf{fix}(g, x : t, e : t')) \wedge \\ &\forall j \leq k, (\varsigma_v, v) \in \mathcal{V}_j[\![t]\!].\ \varsigma_{v'} \star \varsigma_v \text{ defined } \Rightarrow (\varsigma_{v'} \star \varsigma_v, v'\, v) \in \mathcal{C}_j[\![t']\!]\}\end{aligned}$$

$$\begin{aligned}\mathcal{C}_k[\![t]\!] = \{(\varsigma_s, e_s) \mid \forall\, j < k, \sigma_r.\ &\varsigma_s \star \varsigma_r \text{ defined } \Rightarrow \langle \sigma_s + \sigma_r, e_s \rangle \to^j \mathbf{err}\ \vee \exists \sigma_f, e_f. \\ &\langle \sigma_s + \sigma_r, e_s \rangle \to^j \langle \sigma_f + \sigma_r, e_f \rangle \wedge (e_f \text{ is a value } \Rightarrow (\varsigma_f \star \varsigma_r, e_f) \in \mathcal{V}_{k-j}[\![t]\!])\}\end{aligned}$$

$$\mathcal{I}_k[\![\cdot]\!]\theta = \{[]\}$$

$$\mathcal{I}_k[\![\Delta, x : t]\!]\theta = \{\delta[x \mapsto v_x] \mid \delta \in \mathcal{I}_k[\![\Delta]\!]\theta \wedge (\emptyset, v_x) \in \mathcal{V}_k[\![\theta(t)]\!]\}$$

$$\mathcal{L}_k[\![\cdot]\!]\theta = \{(\emptyset, [])\}$$

$$\mathcal{L}_k[\![\Gamma, x : t]\!]\theta = \{(\varsigma \star \varsigma_x, \gamma[x \mapsto v_x]) \mid (\varsigma, \gamma) \in \mathcal{L}_k[\![\Gamma]\!]\theta \wedge (\varsigma_x, v_x) \in \mathcal{V}_k[\![\theta(t)]\!]\}$$

$$\varsigma \equiv \mathcal{H}[\![\sigma]\!] \equiv \bigstar_{(l,f,m) \in \sigma}[l \mapsto_f m]$$

$$\begin{aligned}{}_k[\![\Theta; \Delta; \Gamma \vdash e : t]\!] = \forall \theta, \delta, \gamma, \sigma.\ &\mathrm{dom}(\Theta) = \mathrm{dom}(\theta) \wedge (\varsigma, \gamma) \in \mathcal{L}_k[\![\Gamma]\!]\theta \wedge \delta \in \mathcal{I}_k[\![\Delta]\!]\theta \Rightarrow \\ &(\varsigma, \gamma(\delta(e))) \in \mathcal{C}_k[\![\theta(t)]\!]\end{aligned}$$

4 Proofs

4.1 Lemmas

4.1.1 $\forall \sigma_s, \sigma_r, e.\ \varsigma_s \star \varsigma_r$ defined $\Rightarrow \forall n.\ \langle \sigma_s, e \rangle \rightarrow^n = \langle \sigma_s + \sigma_r, e \rangle \rightarrow^n$

SUFFICES: By induction on n, consider only the cases $\langle \sigma_s, e \rangle \rightarrow \langle \sigma_f, e_f \rangle$ where $\sigma_s \neq \sigma_f$.

PROOF SKETCH: Only OP_{FREE, MATRIX, SHARE, UNSHARE_EQ, GEMM_MATCH} change the heap: the rest are either parametric in the heap or step to an **err**.

PROVE: $\langle \sigma_s + \sigma_r, e \rangle \rightarrow \langle \sigma_f + \sigma_r, e_f \rangle$.

$\langle 1 \rangle 1$. CASE: OP_FREE, $\sigma_s \equiv \sigma' + \{l \mapsto_1 m\}$, $\sigma_f = \sigma'$.
PROOF: Instantiate OP_FREE with $(\sigma' + \sigma_r) + \{l \mapsto_1 m\}$,
valid because $l \notin \mathrm{dom}(\varsigma_r)$ by $\varsigma' \star [l \mapsto_1 m] \star \varsigma_r$ defined (assumption).

$\langle 1 \rangle 2$. CASE: OP_MATRIX
PROOF: Rule has no requirements on σ_s so will also work with $\sigma_s + \sigma_r$.

$\langle 1 \rangle 3$. CASE: OP_SHARE, $\sigma_s \equiv \sigma' + \{l \mapsto_f m\}$, $\sigma_f = \sigma' + \{l \mapsto_{\frac{1}{2} \cdot f} m\} + \{l \mapsto_{\frac{1}{2} \cdot f} m\}$.
PROOF: Union-ing σ_r does not remove $l \mapsto_f m$, so that can be split out of $\sigma_s + \sigma_r$ as before.

$\langle 1 \rangle 4$. CASE: OP_UNSHARE_EQ, $\sigma_s \equiv \sigma' + \{l \mapsto_{\frac{1}{2} \cdot f} m\} + \{l \mapsto_{\frac{1}{2} \cdot f} m\}$, $\sigma_f = \sigma' + \{l \mapsto_f m\}$.

$\langle 2 \rangle 1$. Union-ing σ_r does not remove $l \mapsto_{\frac{1}{2} \cdot f} m$, so that can still be split out of $\sigma_s + \sigma_r$.

$\langle 2 \rangle 2$. There may also be other valid splits introduced by σ_r.

$\langle 2 \rangle 3$. However, by assumption of $\varsigma_s \star \varsigma_r$ defined, any splitting of $\sigma_s + \sigma_r$ will satisfy $f \leq 1$.

$\langle 1 \rangle 5$. CASE: OP_GEMM_MATCH

$\langle 2 \rangle 1$. By assumption of $\varsigma_s \star \varsigma_r$ defined, either l_1 (or l_2, or both) are not in σ_r, or they are and the matrix values they point to are the same.

$\langle 2 \rangle 2$. The permissions (of l_1 and/or l_2) may differ, but OP_GEMM_MATCH universally quantifies over them and leaves them unchanged, so they are irrelevant.

$\langle 2 \rangle 3$. Only the pointed to matrix value at l_3 changes.

$\langle 2 \rangle 4$. SUFFICES: $l_3 \notin \pi_1[\sigma_r]$.

$\langle 2 \rangle 5$. By assumption of $\varsigma_s \star \varsigma_r$ defined, $l_3 \notin \mathrm{dom}(\varsigma_r)$.

$\langle 2 \rangle 6$. Hence $l_3 \notin \pi_1[\sigma_r]$.

4.1.2 $\forall k, t.\ \mathcal{V}_k[\![t]\!] \subseteq \mathcal{C}_k[\![t]\!]$

Follows from definition of $\mathcal{C}_k[\![t]\!]$, $\rightarrow^j$ $(\forall n.\ \langle \sigma, v \rangle \rightarrow^n \langle \sigma, v \rangle)$ for arbitrary $j \leq k$ and 4.1.1.

4.1.3 $\forall \delta, \gamma, v.\ \delta(\gamma(v))$ is a value.

By construction, δ and γ only map variables to values, and values are closed under substitution.

4.1.4 $\forall k, \sigma, \sigma', e, e', t.\ (\varsigma', e') \in \mathcal{C}_k[\![t]\!] \wedge \langle \sigma, e\rangle \to \langle \sigma', e'\rangle \Rightarrow (\varsigma, e) \in \mathcal{C}_{k+1}[\![t]\!]$

ASSUME: arbitrary $j < k + 1$, and σ_r such that $\varsigma \star \varsigma_r$ defined.

$\langle 1\rangle 1$. CASE: $j = 0$. Clearly $\sigma_f = \sigma_s + \sigma_r$ and $e' = e$.
Remains to show that if e is a value then $(\varsigma_s \star \varsigma_r, e) \in \mathcal{V}_k[\![t]\!]$.
This is true vacuously, because by assumption, e is not a value.

$\langle 1\rangle 2$. CASE: $j \geq 1$. We have $\langle \sigma, e\rangle \to^j = \langle \sigma', e'\rangle \to^{j-1}$.
Instantiate $(\varsigma', e') \in \mathcal{C}_k[\![t]\!]$, with $j - 1 < k$ and σ_r to conclude the required conditions.

4.1.5 $j \leq k \Rightarrow {\_}_k[\![\cdot]\!] \subseteq {\_}_j[\![\cdot]\!]$

Lemma 4.1.4 is the inductive step for this lemma for the $\mathcal{C}[\![]\!]$ case.
Need to prove for $\mathcal{V}[\![]\!]$, by induction on t and then index.

SUFFICES: Consider only $t \multimap t'$ case, rest use k directly on structure of type.
ASSUME: Arbitrary $j \leq k$ and $(\varsigma_{v'}, v') \in \mathcal{V}_k[\![t \multimap t']\!]$.
PROVE: $(\varsigma_{v'}, v') \in \mathcal{V}_j[\![t \multimap t']\!]$.

$\langle 1\rangle 1$. v' is of the correct syntactic form (lambda or fixpoint) by assumption.

$\langle 1\rangle 2$. ASSUME: arbitrary $j' \leq j$ and $(\varsigma_v, v) \in \mathcal{V}_{j'}[\![t]\!]$ such that $\varsigma_{v'} \star \varsigma_v$ is defined.

$\langle 1\rangle 3$. SUFFICES: to show $(\varsigma_{v'} \star \varsigma_v, v'v) \in \mathcal{C}_{j'}[\![t']\!]$.

$\langle 1\rangle 4$. This is true by instantiating $(\varsigma_{v'}, v') \in \mathcal{V}_k[\![t \multimap t']\!]$ with $j' \leq k$ and $(\varsigma_v, v) \in \mathcal{V}_{j'}[\![t]\!]$.

4.1.6 $\forall \Delta, \Gamma, t, k, \theta, \delta, \gamma.\ \delta \in \mathcal{I}_k[\![\Delta]\!]\theta \wedge \gamma \in \pi_2[\mathcal{L}_k[\![\Gamma]\!]\theta] \Rightarrow \mathrm{dom}(\Delta) = \mathrm{dom}(\delta)$ **and** $\mathrm{dom}(\Gamma) = \mathrm{dom}(\gamma)$

PROOF: By induction on Δ and Γ.

4.1.7 $\forall k, \Gamma, \Gamma', \theta, \sigma_+, \gamma_+.\ (\varsigma_+, \gamma) \in \mathcal{L}_k[\![\Gamma, \Gamma']\!]\theta \wedge \Gamma, \Gamma'$ **disjoint** $\Rightarrow$
$\exists \sigma, \gamma, \sigma', \gamma'.\ \sigma_+ = \sigma + \sigma' \wedge \gamma, \gamma'$ **disjoint** $\wedge\, \gamma_+ = \gamma \cup \gamma'$
$\wedge\, (\varsigma, \gamma) \in \mathcal{L}_k[\![\Gamma]\!] \wedge (\varsigma', \gamma') \in \mathcal{L}_k[\![\Gamma']\!]$

PROOF: By induction on Γ'.

4.2 Soundness

$$\forall \Theta, \Delta, \Gamma, e, t.\ \Theta; \Delta; \Gamma \vdash e : t \Rightarrow \forall k.\ {}_k[\![\Theta; \Delta; \Gamma \vdash e : t]\!]$$

Proof sketch: Induction over the typing judgements.

Assume: 1. Arbitrary $\Theta, \Delta, \Gamma, e, t$ such that $\Theta; \Delta; \Gamma \vdash e : t$.
2. Arbitrary $k, \theta, \delta, \gamma, \sigma$ such that:
a. $\mathrm{dom}(\Theta) = \mathrm{dom}(\theta)$
b. $(\varsigma, \gamma) \in \mathcal{L}_k[\![\Gamma]\!]\theta$
c. $\delta \in \mathcal{I}_k[\![\Delta]\!]\theta$.
3. W.l.o.g., all variables are distinct,
hence $\mathrm{dom}(\Delta)$ and $\mathrm{dom}(\Gamma)$ are disjoint
so $\gamma \circ \delta = \delta \circ \gamma$ (as substitutions defined recursively over expressions).

Prove: $(\varsigma, \gamma(\delta(e))) \in \mathcal{C}_k[\![\theta(t)]\!]$.
Assume: Arbitrary $j < k$ and σ_r, such that $\varsigma \star \varsigma_r$ defined.
Suffices: $\langle \sigma + \sigma_r, e \rangle \to^j \mathbf{err}\ \vee \exists \sigma_f, e_f.\ \langle \sigma + \sigma_r, e \rangle \to^j \langle \sigma_f + \sigma_r, e_f \rangle$
$\wedge (e_f$ is a value $\Rightarrow (\varsigma_f \star \varsigma_r, e_f) \in \mathcal{V}_{k-j}[\![t]\!])$.
Suffices: By 4.1.1, to show $\langle \sigma, e \rangle \to^j \mathbf{err}\ \vee \exists \sigma_f, e_f.\ \langle \sigma, e \rangle \to^j \langle \sigma_f, e_f \rangle$
$\wedge (e_f$ is a value $\Rightarrow (\varsigma_f, e_f) \in \mathcal{V}_{k-j}[\![t]\!])$

$\langle 1 \rangle 1$. Case: Ty_Let.

$\langle 2 \rangle 1$. By induction,
1. $\forall k.\ {}_k[\![\Theta; \Delta; \Gamma \vdash e : t]\!]$
2. $\forall k.\ {}_k[\![\Theta; \Delta; \Gamma', x : t \vdash e' : t']\!]$.

$\langle 2 \rangle 2$. By 2b, 3 and 4.1.7, we know there exists the following:
1. $(\varsigma_e, \gamma_e) \in \mathcal{L}_k[\![\Gamma]\!]$
2. $\gamma = \gamma_e \cup \gamma_{e'}$
3. $\sigma = \sigma_e + \sigma_{e'}$.

$\langle 2 \rangle 3$. So, using $k, \theta, \delta, \gamma_e, \sigma_e$, we have $(\varsigma_e, \gamma_e(\delta(e))) \in \mathcal{C}_k[\![\theta(t)]\!]$.

$\langle 2 \rangle 4$. By $\langle 2 \rangle 2$, $(\varsigma_e, \gamma(\delta(e))) \in \mathcal{C}_k[\![\theta(t)]\!]$.

$\langle 2 \rangle 5$. By definition of $\mathcal{C}_k[\![\cdot]\!]$ and $\langle 2 \rangle 2$, we instantiate with j and $\sigma_r = \sigma_{e'}$ to conclude that $\langle \varsigma, \gamma(\delta(e)) \rangle$ either takes j step to **err** or another heap-and-expression $\langle \sigma_f, \gamma(\delta(e_f)) \rangle$.

$\langle 2 \rangle 6$. Case: **err**
By Op_Context_Err, the whole expression reduces to **err** in $j < k$ steps.

$\langle 2 \rangle 7$. Case: j steps to another heap and expression.
By Op_Context and, the whole expression does the same.

$\langle 2 \rangle 8$. If it is not a value, we are done. If it is $(\varsigma_f, v) \in \mathcal{V}_{k-j}[\![\theta(t)]\!]$ by 4.1.3.
Suffices: $(\varsigma_f \star \varsigma_{e'}, \mathbf{let}\, x = v\, \mathbf{in}\, \gamma(\delta(e'))) \in \mathcal{C}_{k-j}[\![\theta(t')]\!]$ by 4.1.4 j times.
Suffices: $(\varsigma_f \star \varsigma_{e'}, \gamma(\delta(e'))[x/v]) \in \mathcal{C}_{k-j-1}[\![\theta(t')]\!]$ by 4.1.4.

$\langle 2 \rangle 9$. By 4.1.5, $(\varsigma_{e'}, \gamma_{e'}[x \mapsto v]) \in \mathcal{L}_k[\![\Gamma', x : t]\!]\theta \subseteq \mathcal{L}_{k-j-1}[\![\Gamma', x : t]\!]\theta$.

$\langle 2 \rangle 10$. Instantiate 2 of step $\langle 2 \rangle 1$ with $\theta, k-j-1, \delta, \gamma_{e'}[x \mapsto v], \sigma_{e'}$ to conclude
$(\varsigma_{e'}, \gamma_{e'}[x \mapsto v](\delta(e'))) \in \mathcal{C}_{k-j-1}[\![\theta(t')]\!]$.

⟨2⟩11. By 3, we have $\gamma(\delta(e'))[x/v] = \gamma_{e'}[x \mapsto v](\delta(e'))$ and
by 4.1.1 we conclude $(\varsigma_f \star \varsigma_{e'}, \gamma(\delta(e'))[x/v]) \in \mathcal{C}_{k-j-1}[\![\theta(t')]\!]$

⟨1⟩2. CASE: TY_PAIR_ELIM.
PROOF SKETCH: Similar to TY_LET, but with the following key differences.

⟨2⟩1. When $(\varsigma_f, v) \in \mathcal{V}_{k-j}[\![\theta(t_1) \otimes \theta(t_2)]\!]$, we have $v = (v_1, v_2)$.

⟨2⟩2. SUFFICES: $(\varsigma_{e'}, \gamma(\delta(e'))) \in \mathcal{C}_{k-j-1}[\![\theta(t')]\!]$ by 4.1.4 $j+1$ times.

⟨2⟩3. By 4.1.5, $(\varsigma_{e'}, \gamma_{e'}[a \mapsto v_1, b \mapsto v_2]) \in \mathcal{L}_k[\![\Gamma', a : t_1, b : t_2]\!]\theta \subseteq \mathcal{L}_{k-j-1}[\![\Gamma', a : t_1, b : t_2]\!]\theta$.

⟨2⟩4. Instantiate ${}_{k-j-1}[\![\Theta; \Delta; \Gamma', a : t_1, b : t_2 \vdash e' : t']\!]$ with $\theta, \delta, \gamma_{e'}[a \mapsto v_1, b \mapsto v_2], \sigma_{e'}$.

⟨2⟩5. By 3 $(\varsigma_{e'}, \gamma(\delta(e'))) \in \mathcal{C}_{k-j-1}[\![\theta(t')]\!]$.

⟨1⟩3. CASE: TY_BANG_ELIM.
PROOF SKETCH: Similar to TY_LET, but with the following key differences.

⟨2⟩1. When $(\varsigma_f, v) \in \mathcal{V}_{k-j}[\![\theta(!t)]\!]$, since $\mathcal{V}_{k-j}[\![\theta(!t)]\!] = \mathcal{V}_{k-j}[\![!\theta(t)]\!]$,
we have $\varsigma_f = \emptyset$ and $v = \textbf{Many}\, v'$ for some $(\emptyset, v') \in \mathcal{V}_{k-j}[\![\theta(t)]\!]$.

⟨2⟩2. SUFFICES: $(\varsigma_{e'}, \textbf{let Many}\, x \;=\; \textbf{Many}\, v' \,\textbf{in}\, \gamma(\delta(e'))) \in \mathcal{C}_{k-j}[\![\theta(t)]\!]$.

⟨2⟩3. SUFFICES: $(\varsigma_{e'}, \gamma(\delta(e'))[x/v]) \in \mathcal{C}_{k-j-1}[\![\theta(t)]\!]$ by 4.1.4 $j+1$ times.

⟨2⟩4. Instantiate ${}_{k-j-1}[\![\Theta; \Delta, x : t, \Gamma' \vdash e' : t']\!]$ with $\theta, \delta_{e'} = \delta[x \mapsto v'], \gamma_{e'}, \sigma_{e'}$.

⟨2⟩5. By 3, $(\varsigma_{e'}, \gamma(\delta(e'))[x/v]) \in \mathcal{C}_{k-j-1}[\![\theta(t)]\!]$.

⟨1⟩4. CASE: TY_UNIT_ELIM.
PROVE: $(\sigma, \gamma(\delta(\textbf{let}\,() \;=\; e\, \textbf{in}\, e'))) \in \mathcal{C}_k[\![\theta(t)]\!]$.
PROOF: Similar to TY_LET but with OP_LET_UNIT.

⟨2⟩1. When $(\sigma_{ef}, v) \in \mathcal{V}_{k-j}[\![\textbf{unit}]\!]$, we have $\sigma_{ef} = \emptyset$ and $v = ()$.

⟨2⟩2. SUFFICES: $(\sigma_{e'}, \gamma(\delta(e'))) \in \mathcal{C}_{k-j-1}[\![\theta(t')]\!]$ by 4.1.4.

⟨2⟩3. DEFINE: $\gamma_{e'}$ to be the restriction of γ to $\mathrm{dom}(\Gamma')$.
Thus, by 4.1.5, $(\sigma_{e'}, \gamma_{e'}) \in \mathcal{L}_k[\![\Gamma']\!]\theta \subseteq \mathcal{L}_{k-j-1}[\![\Gamma']\!]\theta$.

⟨2⟩4. Instantiate $[\![\Theta; \Delta; \Gamma' \vdash e' : t']\!]$ with $\theta, k-j-1, \delta, \gamma_{e'}, \sigma_{e'}$.

⟨2⟩5. By 3 $(\sigma_{e'}, \gamma(\delta(e'))) \in \mathcal{C}_{k-j-1}[\![\theta(t')]\!]$.

⟨1⟩5. CASE: TY_BOOL_ELIM.
PROVE: $(\sigma, \gamma(\delta(\textbf{if}\, e\, \textbf{then}\, e_1\, \textbf{else}\, e_2))) \in \mathcal{C}_k[\![\theta(t)]\!]$.
PROOF: Similar to TY_UNIT_ELIM but with OP_IF_{TRUE,FALSE}
and $\sigma_{ef} = \emptyset$ and $v = \textbf{Many true}$ or $v = \textbf{Many false}$.

⟨1⟩6. CASE: TY_BANG_INTRO.
PROVE: $(\sigma, \gamma(\delta(\textbf{Many}\, e))) \in \mathcal{C}_k[\![\theta(!t)]\!]$.
SUFFICES: $(\sigma, \textbf{Many}\, \gamma(\delta(e))) \in \mathcal{C}_k[\![!\theta(t)]\!]$.

⟨2⟩1. By assumption of TY_BANG_INTRO, $e = v$ for some value $v \neq l$, $\Gamma = \emptyset$ and so
$[\![\Theta; \Delta; \cdot \vdash v : t]\!]$ by induction.

$\langle 2\rangle 2.$ SUFFICES: $(\emptyset, \textbf{Many}\, \delta(v)) \in \mathcal{C}_k[\![!\theta(t)]\!]$ by 3 and 2b.

$\langle 2\rangle 3.$ Instantiate $[\![\Theta; \Delta; \cdot \vdash v : t]\!]$ with $\theta, k, \delta, \gamma = [], \sigma = \emptyset$ to obtain $(\emptyset, \delta(v)) \in \mathcal{C}_k[\![\theta(t)]\!]$.

$\langle 2\rangle 4.$ Instantiate $(\emptyset, \delta(v)) \in \mathcal{C}_k[\![\theta(t)]\!]$ with $j = 0$, and $\sigma_r = \emptyset$, to conclude $(\emptyset, v) \in \mathcal{V}_k[\![\theta(t)]\!]$.

$\langle 2\rangle 5.$ By definition of $\mathcal{V}_k[\![!\theta(t)]\!]$, 4.1.3 and 4.1.2 we have $(\emptyset, \textbf{Many}\, \delta(v)) \in \mathcal{C}_k[\![!\theta(t)]\!]$.

$\langle 1\rangle 7.$ CASE: TY_PAIR_INTRO.
PROVE: $(\sigma, \gamma(\delta(\,(e, e')\,))) \in \mathcal{C}_k[\![\theta(t \otimes t')]\!]$.
ASSUME: Arbitrary $j \leq k$ and σ_r.
SUFFICES: Show whole expression either reduces to **err** or a heap and expression in j steps.

$\langle 2\rangle 1.$ DEFINE: $(\sigma_1, \gamma_1) \in \mathcal{L}_j[\![\Gamma]\!]$ similar to (σ_e, γ_e) in TY_LET.

$\langle 2\rangle 2.$ By induction,
1. $[\![\Theta; \Delta; \Gamma_1 \vdash e_1 : t_1]\!]$
2. $[\![\Theta; \Delta; \Gamma_2 \vdash e_2 : t_2]\!]$.

$\langle 2\rangle 3.$ Instantiate the first with $\theta, k, \delta, \gamma_1, \sigma_1$.

$\langle 2\rangle 4.$ Therefore, $(\sigma_1, \gamma_1(\delta(e_1))) \in \mathcal{C}_k[\![\theta(t)]\!]$.

$\langle 2\rangle 5.$ So, $(\sigma_1 \star \sigma_2, \gamma_1(\delta(e_1)))$ either reduces to **err** or a heap and expression in j steps.

$\langle 2\rangle 6.$ CASE: **err**
By OP_CONTEXT_ERR and 3, so too does the whole expression. Since $j \leq k$ and σ_r (for 4.1.1) are arbitrary, $(\sigma, \gamma(\delta(\,(e, e')\,))) \in \mathcal{C}_k[\![\theta(t \otimes t')]\!]$.

$\langle 2\rangle 7.$ CASE: j steps to another heap and expression.
By OP_CONTEXT and 3, the whole expression does the same.

$\langle 2\rangle 8.$ If it is not a value, we are done. If it is $(\sigma_{1f}, v_1) \in \mathcal{V}_{k-j}[\![\theta(t_1)]\!]$ by 4.1.3.
SUFFICES: By 4.1.4, $(\sigma_{1f} \star \sigma_{e_2}, \,(v_1, e_2)\,) \in \mathcal{C}_{k-j}[\![\theta(t_1 \otimes t_2)]\!]$.

$\langle 2\rangle 9.$ Instantiate the second IH with $\theta, j, \delta, \gamma_2, \sigma_2$ defined as per usual.

$\langle 2\rangle 10.$ So, $(\sigma_{1f} \star \sigma_2, \gamma_2(\delta(e_2)))$ either reduces to **err** or a heap and expression in j steps.

$\langle 2\rangle 11.$ CASE: **err**
By OP_CONTEXT_ERR, 3, so too does the whole expression. Since $j \leq k$ and σ_r (for 4.1.1) are arbitrary, $(\sigma_{e_2}, \,(v_1, e_2)\,) \in \mathcal{C}_{k-j}[\![\theta(t_1 \otimes t_2)]\!]$.

$\langle 2\rangle 12.$ CASE: j steps to another heap and expression.
By OP_CONTEXT and 3, the whole expression does the same.

$\langle 2\rangle 13.$ If it is not a value, we are done. If it is $(\sigma_{2f}, v_2) \in \mathcal{V}_{k-j}[\![\theta(t_2)]\!]$ by 4.1.3.
SUFFICES: By 4.1.4, $(\sigma_{1f} \star \sigma_{2f}, \,(v_1, v_2)\,) \in \mathcal{C}_{k-2j}[\![\theta(t_1 \otimes t_2)]\!]$.

$\langle 2\rangle 14.$ By 4.1.5 and 4.1.2, $(\sigma_{1f} \star \sigma_{2f}, \,(v_1, v_2)\,) \in \mathcal{V}_{k-j}[\![\cdot]\!] \subseteq \mathcal{V}_{k-2j}[\![\cdot]\!] \subseteq \mathcal{C}_{k-2j}[\![\cdot]\!]$ as needed.

$\langle 1\rangle 8.$ CASE: TY_LAMBDA.
PROVE: $(\sigma, \gamma(\delta(\textbf{fun}\, x : t \to e))) \in \mathcal{C}_k[\![\theta(t \multimap t')]\!]$.
SUFFICES: By 6, to show $\ldots \in \mathcal{V}_k[\![\theta(t \multimap t')]\!]$.
ASSUME: Arbitrary $j < k$, $(\sigma_v, v) \in \mathcal{V}_j[\![\theta(t)]\!]$ such that $\sigma \star \sigma_v$ is defined.
SUFFICES: $(\sigma \star \sigma_v, \gamma(\delta(\textbf{fun}\, x : t \to e))\, v) \in \mathcal{C}_j[\![\theta(t')]\!]$.

SUFFICES: $(\sigma \star \sigma_v, \gamma(\delta(e))[x/v]) \in \mathcal{C}_j[\![\theta(t')]\!]$.

⟨2⟩1. By induction, $[\![\Theta; \Delta; \Gamma, x : t \vdash e]\!]$.

⟨2⟩2. Instantiate it $\theta, j-1, \gamma[x \mapsto v], \sigma_v \star \sigma$.

⟨2⟩3. Hence, $(\sigma_v \star \sigma, \gamma[x \mapsto v](\delta(e))) \in \mathcal{C}_{j-1}[\![\theta(t)]\!]$.

⟨2⟩4. By 3, we are done.

⟨1⟩9. CASE: TY_APP.
PROVE: $(\sigma, \gamma(\delta(\,e\,e'\,))) \in \mathcal{C}_k[\![\theta(t)]\!]$.
ASSUME: Arbitrary j and σ_r such that $\sigma \star \sigma_r$ defined.
SUFFICES: Show whole expression either reduces to **err** or a heap and expression in j steps.

⟨2⟩1. By induction,
1. $[\![\Theta; \Delta; \Gamma \vdash e : t' \multimap t]\!]$
2. $[\![\Theta; \Delta; \Gamma' \vdash e' : t']\!]$.

⟨2⟩2. Instantiate the first with $\theta, k, \delta, \gamma_e, \sigma_e$ as per usual definitions, to conclude $(\sigma_e, \gamma_e(\delta(e))) \in \mathcal{C}_k[\![\theta(t' \multimap t)]\!]$.

⟨2⟩3. Instantiate *this* with j and $\sigma_{e'}$ to conclude $(\sigma = \sigma_e \star \sigma_{e'}, \gamma(\delta(\,e\,e'\,)))$ reduces to **err** or another heap and expression in j steps (using 3).

⟨2⟩4. CASE: **err**
By OP_CONTEXT_ERR, so too does the whole expression.
Since $j \leq k$ and σ_r (for 4.1.1) are arbitrary, $(\sigma, \gamma(\delta(\,e\,e'\,))) \in \mathcal{C}_k[\![\theta(t' \multimap t)]\!]$.

⟨2⟩5. CASE: j steps to another heap and expression.
By OP_CONTEXT, the whole expression does the same.
If it is not a value, we are done.
If it is $(\sigma_{ef}, \mathbf{fun}\, x : t \to e_b) \in \mathcal{V}_{k-j}[\![\theta(t' \multimap t)]\!]$ by 4.1.3.

⟨2⟩6. SUFFICES: By 4.1.4, to show $(\sigma_{ef} \star \sigma_{e'}, \gamma(\delta(\,(\mathbf{fun}\, x : t \to e_b)\, e'))) \in \mathcal{C}_{k-j}[\![\theta(t)]\!]$.

⟨2⟩7. Instantiate the second IH with $\theta, j, \delta, \gamma_{e'}, \sigma_{e'}$ defined as per usual.

⟨2⟩8. So, $(\sigma_{ef} \star \sigma_{e'}, \gamma_{e'}(\delta(e')))$ either reduces to **err** or a heap and expression in j steps.

⟨2⟩9. CASE: **err**
By OP_CONTEXT_ERR and 3, so too does the whole expression. Since $j \leq k$ and σ_r (for 4.1.1) are arbitrary, $(\sigma_{ef} \star \sigma_{e'}, \gamma(\delta(\,(\mathbf{fun}\, x : t \to e_b)\, e'))) \in \mathcal{C}_{k-j}[\![\theta(t)]\!]$.

⟨2⟩10. CASE: j steps to another heap and expression.
By OP_CONTEXT and 3, the whole expression does the same.

⟨2⟩11. If it is not a value, we are done. If it is, by definition of $(\sigma_{ef}, \mathbf{fun}\, x : t \to e_b) \in \mathcal{V}_{k-j}[\![\theta(t' \multimap t)]\!]$, we have $(\sigma_{ef} \star \sigma_{e'f}, \gamma(\delta(\,(\mathbf{fun}\, x : t \to e_b)\, v'))) \in \mathcal{C}_{k-2j}[\![\theta(t)]\!]$.

⟨1⟩10. CASE: TY_GEN.
PROVE: $(\sigma, \gamma(\delta(\mathbf{fun}\, fc \to e))) \in \mathcal{C}_k[\![\theta(\forall\, fc.\, t)]\!]$.

⟨1⟩11. CASE: TY_SPC.

PROVE: $(\sigma, \gamma(\delta(e\,[f]))) \in \mathcal{C}_k[\![\theta(t\,[fc/f])]\!]$.

$\langle 1\rangle$12. CASE: TY_FIX.
PROVE: $(\sigma, \gamma(\delta(\mathbf{fix}(g, x : t, e : t')))) \in \mathcal{C}_k[\![\theta(!(t \multimap t'))]\!]$.
SUFFICES: to show $\ldots \in \mathcal{V}_k[\![!(\theta(t) \multimap \theta(t'))]\!]$, by 4.1.2.

$\langle 2\rangle$1. ASSUME: Arbitrary $j < k$ and $(\sigma, v) \in \mathcal{V}_j[\![\theta(t)]\!]$.

$\langle 2\rangle$2. SUFFICES: $(\sigma, \mathit{letManyG}\ g\, v) \in \mathcal{C}_j[\![\theta(t')]\!]$.

$\langle 2\rangle$3. LET: $e_1 = e[g/\mathbf{fun}\, x : t \to \mathit{letManyG}\ g\, x]$.

$\langle 2\rangle$4. SUFFICES: by 4.1.4, $(\sigma, (\mathbf{fun}\, x : t \to e_1)\, v) \in \mathcal{C}_{j-1}[\![\theta(t')]\!]$.

$\langle 2\rangle$5. SUFFICES: by 4.1.4, $(\sigma, e_1[x/v]) \in \mathcal{C}_{j-2}[\![\theta(t')]\!]$.

$\langle 2\rangle$6. By induction, we have $[\![\Theta; \Delta, g : t \multimap t'; x : t \vdash e : t']\!]$.

$\langle 2\rangle$7. Instantiate this with $\theta, j-2, \delta[g \mapsto \mathbf{fun}\, x : t \to e_1], \gamma = [x \mapsto v], \sigma$ (???).
PROVE: $(\sigma, \mathbf{fun}\, x : t \to e_1) \in \mathcal{V}_{j-2}[\![\theta(t) \multimap \theta(t')]\!]$.

$\langle 3\rangle$1. SUFFICES: by 4.1.4, $(\sigma', e_1[x/v']) \in \mathcal{C}_{j-2}[\![\theta(t')]\!]$ for arbitrary $(\sigma', v') \in \mathcal{V}_{j-2}[\![\theta(t)]\!]$.

$\langle 3\rangle$2. We can again use the induction hypothesis $[\![\Theta; \Delta, g : t \multimap t'; x : t \vdash e : t']\!]$.

$\langle 3\rangle$3. But since it's true for $\mathcal{C}_0[\![\cdot]\!]$ (base case), it's true by induction ???

$\langle 2\rangle$8. Lastly, we show $\delta(\gamma(e)) = e_1[x/v]$, which follows by their definitions, to conclude $(\sigma, e_1[x/v]) \in \mathcal{C}_{j-2}[\![\theta(t')]\!]$.

$\langle 1\rangle$13. CASE: TY_VAR_LIN.
PROVE: $(\sigma, \gamma(\delta(x))) \in \mathcal{C}_k[\![\theta(t)]\!]$.

$\langle 2\rangle$1. $\Gamma = \{x : t\}$ by assumption of TY_VAR_LIN.

$\langle 2\rangle$2. SUFFICES: $(\sigma, \gamma(x)) \in \mathcal{C}_k[\![\theta(t)]\!]$ by 3.

$\langle 2\rangle$3. By 2b, there exist $(\sigma_x, v_x) \in \mathcal{V}_k[\![\theta(t)]\!]$, such that $\sigma = \sigma_x$ and $\gamma = [x \mapsto v_x]$.

$\langle 2\rangle$4. Hence, $(\sigma_x, v_x) \in \mathcal{C}_k[\![\theta(t)]\!]$, by 4.1.2.

$\langle 1\rangle$14. CASE: TY_VAR.
PROVE: $(\sigma, \gamma(\delta(x))) \in \mathcal{C}_k[\![\theta(t)]\!]$.

$\langle 2\rangle$1. $x : t \in \Delta$ and $\Gamma = \emptyset$ by assumption of TY_VAR.

$\langle 2\rangle$2. SUFFICES: $(\emptyset, \delta(x)) \in \mathcal{C}_k[\![\theta(t)]\!]$ by 3 and 2b.

$\langle 2\rangle$3. By 2c, there exists v_x such that $(\emptyset, v_x) \in \mathcal{V}_k[\![\theta(t)]\!]$.

$\langle 2\rangle$4. Hence, $(\emptyset, v_x) \in \mathcal{C}_k[\![\theta(t)]\!]$, by 4.1.2.

$\langle 1\rangle$15. CASE: TY_UNIT_INTRO.
PROVE: $(\sigma, \gamma(\delta(\,()\,))) \in \mathcal{C}_k[\![\theta(\mathbf{unit})]\!]$.

⟨1⟩16. CASE: TY_BOOL_TRUE, TY_BOOL_FALSE, TY_INT_INTRO, TY_ELT_INTRO.
Similar to TY_UNIT_INTRO.

5 Grammar Definition

m	$::=$			matrix expressions
	$\mid$	M		matrix variables
	$\mid$	$m + m'$		matrix addition
	$\mid$	$m\, m'$		matrix multiplication
	$\mid$	(m)	S	
f	$::=$			fractional capability
	$\mid$	fc		variable
	$\mid$	1		whole capability
	$\mid$	$\frac{1}{2}f$		
t	$::=$			linear type
	$\mid$	**unit**		unit
	$\mid$	**bool**		boolean (true/false)
	$\mid$	**int**		63-bit integers
	$\mid$	**elt**		array element
	$\mid$	f **arr**		arrays
	$\mid$	f **mat**		matrices
	$\mid$	$!t$		multiple-use type
	$\mid$	$\forall fc.t$	bind fc in t	frac. cap. generalisation
	$\mid$	$t \otimes t'$		pair
	$\mid$	$t \multimap t'$		linear function
	$\mid$	(t)	S	parentheses
p	$::=$			primitive
	$\mid$	**not**		boolean negation
	$\mid$	$(+)$		integer addition
	$\mid$	$(-)$		integer subtraction
	$\mid$	$(*)$		integer multiplication
	$\mid$	$(/)$		integer division
	$\mid$	$(=)$		integer equality
	$\mid$	$(\langle)$		integer less-than
	$\mid$	$(+.)$		element addition
	$\mid$	$(-.)$		element subtraction
	$\mid$	$(*.)$		element multiplication
	$\mid$	$(/.)$		element division
	$\mid$	$(=.)$		element equality
	$\mid$	$(<.)$		element less-than
	$\mid$	**set**		array index assignment
	$\mid$	**get**		array indexing
	$\mid$	**share**		share array
	$\mid$	**unshare**		unshare array
	$\mid$	**free**		free arrary
	$\mid$	**array**		Owl: make array
	$\mid$	**copy**		Owl: copy array
	$\mid$	**sin**		Owl: map sine over array
	$\mid$	**hypot**		Owl: $x_i := \sqrt{x_i^2 + y_i^2}$
	$\mid$	**asum**		BLAS: $\sum_i \lvert x_i \rvert$

	|	**axpy**		BLAS: $x := \alpha x + y$
	|	**dot**		BLAS: $x \cdot y$
	|	**rotmg**		BLAS: see its docs
	|	**scal**		BLAS: $x := \alpha x$
	|	**amax**		BLAS: $\mathrm{argmax}\, i:\ x_i$
	|	**setM**		matrix index assignment
	|	**getM**		matrix indexing
	|	**shareM**		share matrix
	|	**unshareM**		unshare matrix
	|	**freeM**		free matrix
	|	**matrix**		Owl: make matrix
	|	**copyM**		Owl: copy matrix
	|	**copyM_to**		Owl: copy matrix onto another
	|	**sizeM**		dimension of matrix
	|	**trnsp**		transpose matrix
	|	**gemm**		BLAS: $C := \alpha A^{T?} B^{T?} + \beta C$
	|	**symm**		BLAS: $C := \alpha AB + \beta C$
	|	**posv**		BLAS: Cholesky decomp. and solve
	|	**potrs**		BLAS: solve with given Cholesky
v	$::=$			values
	|	p		primitives
	|	x		variable
	|	$()$		unit introduction
	|	**true**		true
	|	**false**		false
	|	k		integer
	|	$l \cdot f$		heap location
	|	el		array element
	|	**Many** v		!-introduction
	|	**fun** $fc \to v$		frac. cap. abstraction
	|	$v[f]$		frac. cap. specialisation
	|	(v, v')		pair introduction
	|	**fun** $x : t \to e$	bind x in e	abstraction
	|	**fix** $(g, x : t, e : t')$	bind $g \cup x$ in e	fixpoint
	|	(v)	S	parentheses
e	$::=$			expression
	|	p		primitives
	|	x		variable
	|	**let** $x = e$ **in** e'	bind x in e'	let binding
	|	$()$		unit introduction
	|	**let** $() = e$ **in** e'		unit elimination
	|	**true**		true
	|	**false**		false
	|	**if** e **then** e_1 **else** e_2		if
	|	k		integer
	|	$l \cdot f$		heap location

	$\mid$	el		array element
	$\mid$	$\mathbf{Many}\ e$		!-introduction
	$\mid$	$\mathbf{let}\ \mathbf{Many}\ x = e\ \mathbf{in}\ e'$		!-elimination
	$\mid$	$\mathbf{fun}\ fc \to e$		frac. cap. abstraction
	$\mid$	$e[f]$		frac. cap. specialisation
	$\mid$	(e, e')		pair introduction
	$\mid$	$\mathbf{let}\ (a, b) = e\ \mathbf{in}\ e'$	bind $a \cup b$ in e'	pair elimination
	$\mid$	$\mathbf{fun}\ x : t \to e$	bind x in e	abstraction
	$\mid$	$e\ e'$		application
	$\mid$	$\mathbf{fix}\ (g, x : t, e : t')$	bind $g \cup x$ in e	fixpoint
	$\mid$	(e)	S	parentheses
C	$::=$			evaluation contexts
	$\mid$	$\mathbf{let}\ x = [-]\ \mathbf{in}\ e$	bind x in e	let binding
	$\mid$	$\mathbf{let}\ () = [-]\ \mathbf{in}\ e$		unit elimination
	$\mid$	$\mathbf{if}\ [-]\ \mathbf{then}\ e_1\ \mathbf{else}\ e_2$		if
	$\mid$	$\mathbf{Many}\ [-]$		!-introduction
	$\mid$	$\mathbf{let}\ \mathbf{Many}\ x = [-]\ \mathbf{in}\ e$		!-elimination
	$\mid$	$\mathbf{fun}\ fc \to [-]$		frac. cap. abstraction
	$\mid$	$[-][f]$		frac. cap. specialisation
	$\mid$	$([-], e)$		pair introduction
	$\mid$	$(v, [-])$		pair introduction
	$\mid$	$\mathbf{let}\ (a, b) = [-]\ \mathbf{in}\ e$	bind $a \cup b$ in e	pair elimination
	$\mid$	$[-]e$		application
	$\mid$	$v[-]$		application
Θ	$::=$			fractional capability environment
	$\mid$	$\cdot$		
	$\mid$	Θ, fc		
Γ	$::=$			linear types environment
	$\mid$	$\cdot$		
	$\mid$	$\Gamma, x : t$		
	$\mid$	Γ, Γ'		
Δ	$::=$			intuitionistic types environment
	$\mid$	$\cdot$		
	$\mid$	$\Delta, x : t$		
σ	$::=$			heap (multiset of triples)
	$\mid$	$\{\}$		empty heap
	$\mid$	$\sigma + \{l \mapsto_f m_{k_1,k_2}\}$		location l points to matrix m
$StepsTo$	$::=$			result of small step
	$\mid$	$\langle \sigma, e \rangle$		heap and expression
	$\mid$	$\mathbf{err}$		error